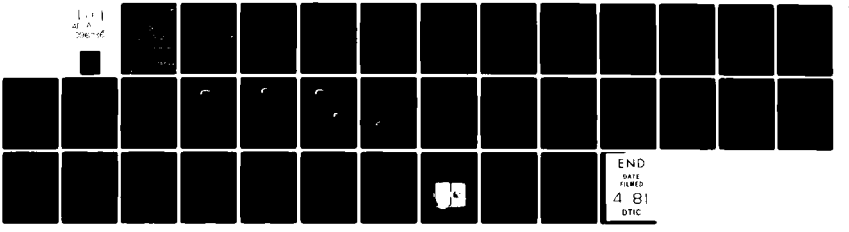
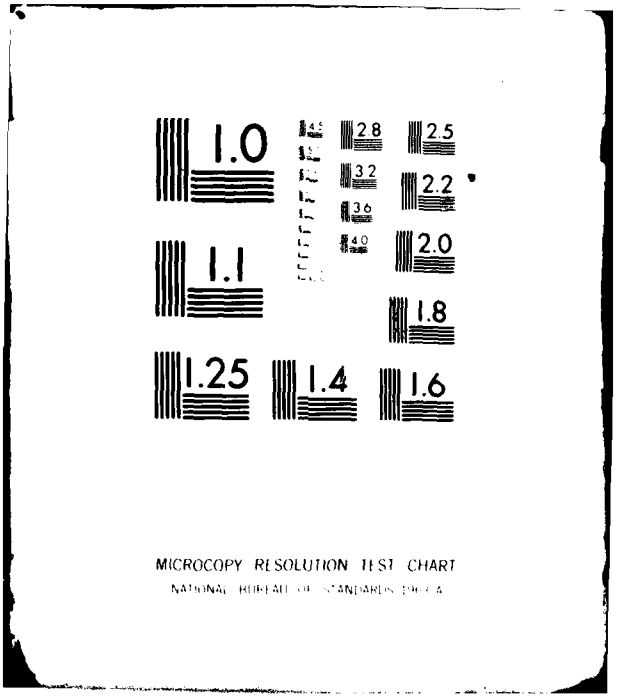
AD-A096 736 NAVAL AIR ENGINEERING CENTER LAKEHURST NJ F/G 9/5
SELF CONTAINED TEST FOR DIGITAL AVIONICS, DESIGNERS GUIDE,(U)
SEP 79 J R SPODOFORA, J E OLEJACK
UNCLASSIFIED NAEC-MISC-92-0369 NL



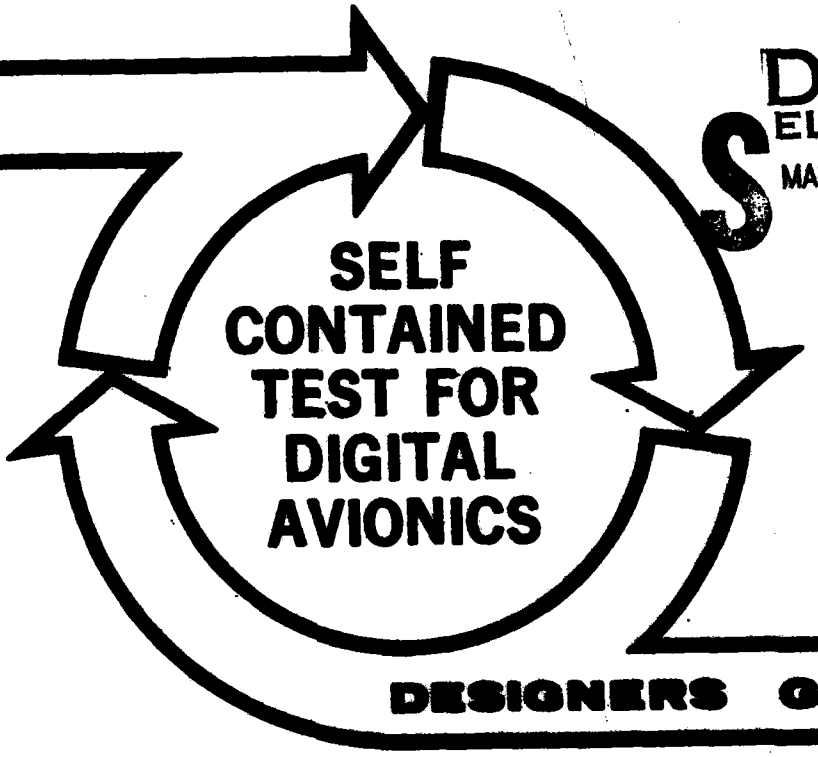
END
DATE FILMED
4 81
DTIC


1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0
MICROCOPY RESOLUTION TEST CHART

LEVEL II

AD A096736

DEPARTMENT OF THE NAVY
NAVAL AIR ENGINEERING CENTER
LAKEHURST, N. J. 08733



DTIC
ELECTE
MAR 24 1981
S D
E

DESIGNERS GUIDE

NAEC MISC 92-0008
1 SEPTEMBER 1978

DTIC FILE COPY

81 3 20 043

NAEC-MISC-92-0369

SELF CONTAINED TEST FOR DIGITAL AVIONICS,

Designers Guide

Prepared by:

J.R. Spodofora
CMR, Inc.

J.E. Olejack
CMR, Inc.

Reviewed by:

J. Bauer 92511
Systems Section

F. Purpura 92511
Head, Systems Section

Approved by:

E.P. McMenamin
Head, Avionics Support
Equipment Division

11-19-79 da

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

FIGURE **PAGE**

SECTION I

AN INTRODUCTION TO SCT

The complexity of VLSI circuits considered for use in future avionics modules makes testing of these modules virtually impossible unless provisions for test are incorporated from the very beginning of design. The circuit complexity and packaging density of VLSI will make designing for testability an important requirement. With the development of a testable design, the equipment can then be maintained in a cost effective manner utilizing Self-Contained-Test (SCT) techniques.

The most direct method of providing complete fault isolation in a digital system is to employ Self-Contained-Test hardware on each replaceable unit (module). This approach is practical if the SCT hardware remains a small percentage of the total cost of the replaceable unit and requires negligible cooling and power. Systems using VLSI technology can meet these requirements as mentioned above.

There are several advantages to the use of SCT in modules containing VLSI circuitry. Since design of the SCT logic is an integral portion of the module design, functional circuitry revisions can systematically incorporate the Self-Contained-Test circuitry, system architecture has minimal effect on SCT, and each module is essentially self testable independent of the system in which it is used.

In future systems, in order to provide for testable designs which require a minimum of hardware and software for test, testability will have to be considered an important design parameter for VLSI systems.

In the design of tactical circuitry it is necessary to consider external test design for each circuit. Very little additional effort would be required to incorporate SCT design in these circuits. It may be possible to incorporate SCT without the need for additional circuitry by making use of available unused circuitry.

THE SCT CONCEPT

Although SCT was developed for complex LSI/VLSI Systems, it is fully applicable to digital systems employing SSI/MSI components. The SCT concept has two major objectives: (1) detecting module faults, and (2) isolating to the faulty module.

The SCT procedure compares module output patterns (obtained by applying to each module, fixed input patterns) to the known responses of fault-free modules. This test technique is commonly used for static testing of digital modules both with and without feedback loops. A module is considered the optimal replaceable unit; therefore, this SCT design concept isolates to that unit.

Each SCT group of modules includes, independently of all other groups of modules, the following test functions: Control and Pattern Generation and Pattern Checking (comparison) as shown in Figure 1-1. One test pattern source and test control source are implemented in each group of modules. Pattern Checking Logic provided on each module generates a code word that represents the state of each module's output and key test points over a test sequence. The resulting code word is compared on each module after a fixed test sequence to a precomputed correct result. This provides a module pass/fail signal.

Fault isolation to a module level is obtained by identifying a specific module that has a "fail" response. Resulting fault detection and isolation to a single module can be 90-95 percent effective.

Figure 1-1 illustrates testing of a group of three modules in a hypothetical non-feedback circuit configuration. Test patterns generated within this group of modules are multiplexed to modules 2 and 3 from module 1 in place of external

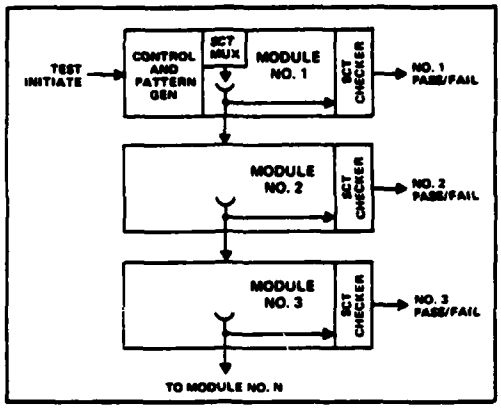


FIGURE 1–1. SCT Fault Detection and Isolation (without feedback) Loops.

operational data. Thus the modules are self-testing in an operational environment. The test data flow is through normal inter-module data paths, from module 1 to 2, and then to 3. An erroneous module output during the test sequence will be detected by the pattern checker on that module. (Note that the pattern checkers do not add delays in module data paths.) In the absence of feedback between modules, fail indication will appear on all modules in sequence after the faulty module. The faulty module is the first module in logic flow order with a "fail" SCT result. Isolation is thus to a single module.

For a hypothetical feedback circuit configuration (Figure 1-2), feedback between modules may propagate a single fault to affect outputs of all modules in the feedback loop, causing multiple "fail" SCT outputs. Since any module in the feedback loop may have initiated the erroneous response, the feedback path must be disabled or broken during SCT in order for fault isolation to a particular module to occur.[1]

Figure 1-2 illustrates the use of one method to break the feedback loop. The multiplexer opens the path of the normal feedback input. With the SCT input selected, all logic flow is from module 1 to module 2, and isolation can be obtained on this basis. The feedback path is re-established after the test patterns are fed through once to check the feedback line. This procedure continues through module N, thereby allowing the isolation to a faulty module to occur.

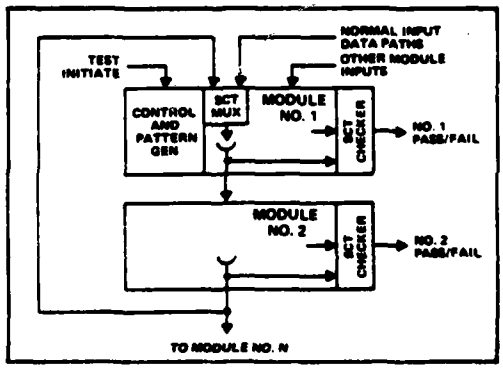


FIGURE 1-2. Isolation with Feedback Loops

Note that these methods isolate to the first faulty module in logic flow order even in the presence of multiple faulty modules. Retest after module replacement then isolates to the next faulty module in logic flow order.

SECTION II

PRELIMINARY SCT GUIDELINES

INTRODUCTION

This section presents, in preliminary form, guidelines for logic design of testable hardware. The guidelines are intended to form the basis of specifications for advanced avionics procurement. These are supplemented by general requirements for implementation of SCT test techniques, trade-offs in SCT application and definitions to provide fixed terminology for test of advanced digital avionics. Application of the guidelines and definitions will supplement and increase the effectiveness of suggested SCT techniques.

In using SCT on tactical hardware, each module shall be able to test itself to a high degree of completeness. The system shall only be required to supply power, cooling, clock signals and a signal to initiate SCT.

Logic design shall adhere to standards in order to provide logic which can be tested and isolated economically, completely and quickly. Each module shall be testable with normal initialization procedures.

HARDWARE DESIGN FOR TESTABILITY

If standards are applied effectively to establish testability in digital designs, the Navy or contractor must have a practical means of establishing conformance. Navy aspects of design evaluation of testability are a matter of subjective engineering judgment and trade-off. However, practical evaluation methods are presented for many of the following areas of design for test. Due to trade-offs of various systems, few practices are disallowed in any form. Rather, desirable practices (and some undesirable ones) are discussed.

Design for test requirements must be established early. Testability should be incorporated as an integral portion of design.

Figure 2-1 illustrates the possible flow of a design with rules for design for testability followed and conformance verified, and a similar procedure for approval of the final design for SCT.

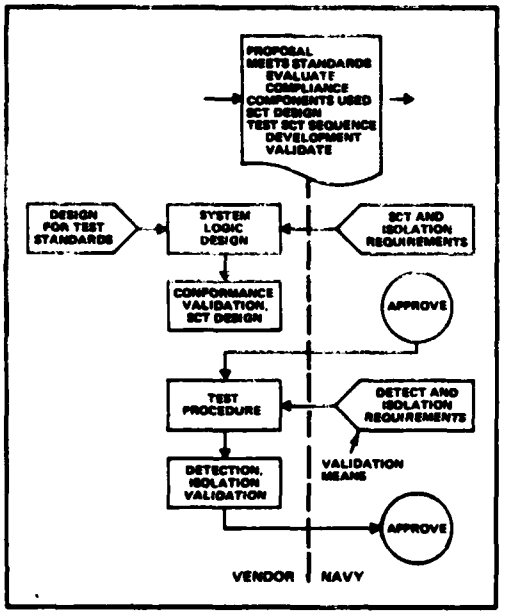


FIGURE 2-1. Design Flow for Testability

INITIALIZATION

Economical testing requires sequential circuits to be initialized prior to testing. The internal memory states will control the outputs and therefore, must be driven to some initial state before testing can begin. This is an important factor in test completeness. Improved and earlier fault detection results from rapid initialization of network memory to known values. Initialization results in shorter test sequences and execution times. The best solution to the above mentioned problem is to incorporate in the logic design, a master reset input, that, when activated, drives the circuit to a known state. Circuits could also be initialized by monitoring the output ter-

minals and applying input patterns until the desired state is reached. This requires a decision making capability in the test circuitry, a costly complication. For some circuit designs, the initialization sequence may be prohibitively lengthy or may not exist.[3]

COUNTER AND SHIFT REGISTER CHAINS

The way in which counters and shift registers are configured greatly affects testabililty, test generation cost and test sequence length. Parallel entry and the ability to hold specific counts in a counter are very desirable features which eliminate the need for long counts in test sequences. Long shift registers should be configured to allow isolation from subsequent logic. Some suggested means of doing this are indicated in Figure 2-2.

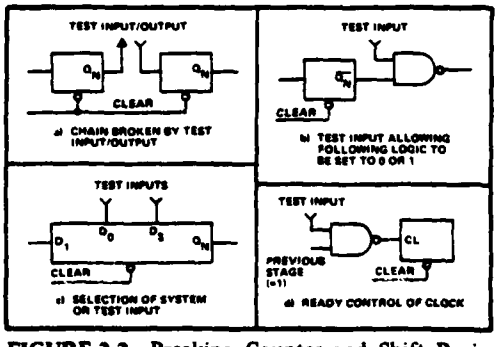


FIGURE 2-2. Breaking Counter and Shift Register Chains

Test data may be obtained without need to be shifted or counted through the entire register as shown in Figure 2-2. Clock inputs should be easily accessible for test, to avoid compounding the sequence length required to transfer data through the chain. As the illustration indicates shift or counter chains may be broken in several ways.

In Figure 2-2a, the shift register or counter chain has been broken by a pair of test inputs/outputs (I/Os).

An alternative method of inserting data into a register or counter chain is shown in Figure 2-2b.

With the test input high, the master clear can be used to immediately provide zero data to following logic. With the test input in the low state, "1" data is input to the following logic. If logic following a long shift register or counter chain is significant, this test hardware is desirable to exercise following logic for "0" and "1" chain output without waiting for propagation through the entire chain.

Figure 2-2c indicates one method which may be used to enter test data using registers with multiple data inputs. Here the data select input (D_S) and one data input (D_O) are test inputs. ORed or ANDed multiple data inputs can also be used, with one input functioning as a test input.

Ready control of a clock input in the center of a counter is illustrated in Figure 2-2d. ORed or ANDed dual clock inputs may also be used for this purpose.

Many of the test inputs as listed above can be used to control test of more than one shift register or counter.

Unused states in counters should be avoided. If such states are necessary, design shall minimize logic to get an operational state. This logic is very difficult to test, as the unused states must be entered for test. As the unused states might be entered only on a probability basis on power up, probability tests only are likely to result.

LOGIC PARTITIONING

Although many factors will influence system partitioning, test considerations should be heavily weighted. Isolation, particulary, is influenced by system partitioning. Functional partitioning and bit slice partitioning, two partitioning philosophies having certain test advantages, are discussed below.

Packaging should partition functionally related (i.e., higher interconnected) circuits onto the same module. This facilitates test

(SCT) for each unit, aids isolation, and reduces unit I/O. A conceptual illustration is given in Figure 2-3a.

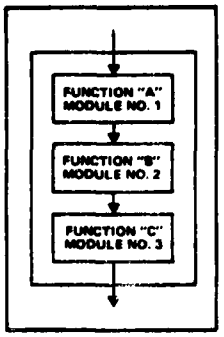


FIGURE 2-3a. Functional Partitioning

Functional partitioning can be applied to indicate partitioning logic on circuits of the same types and purpose onto each hardware unit. Partitioning all arithmetic logic of a computer onto one logic unit is an example of functional partitioning.

Functionally partitioned units provide for systematic test due to their nature. Key delay paths (as an adder carry) can be kept on a single module. Fewer call or device types are needed, and more use is made of complex circuits by concentrating logic for a specific function on each module. Likewise, the concentration of a specific function reduces the need for repetition of distributed control and minimizes modules to which centralized control must fan out.

Bit slice partitioning as shown in Figure 2-3b allows most of the module's logic to be placed on identical or similar modules which are able to perform its processing functions on a limited number of input bits. This contrasts to functional partitioning where a limited subset of processing functions are performed on all input data.

Generally, fewer module I/O are required in a bit slice partition. This results as the bit slice partition is along the path of data flow. Reduction in module outputs will allow reduced checker hardware. Feedback paths in data flow can often be contained in one module with bit slice partitioning. This aids isolation and reduces the need for provision of SCT hardware for feedback control. Isolation problems of busing (or tri-state) logic are also reduced as signals on the same bit position are enabled onto the bus from the same module. This specifically aids isolation for bus (or any driver) stuck at zero (SAO) faults, or multiple signals incorrectly enabled onto the same bus bit.

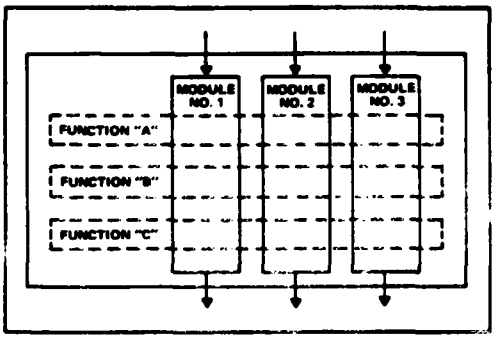


FIGURE 2-3b. Bit Slice Partitioning

Identical bit slice modules can reduce SCT hardware significantly and obtain good isolation by comparison of identical module outputs. Finally, a reduction in module types for identical modules improves logistics problems and test generation.

Test generation cost is reduced if memories are provided on modules having memory hardware only. Efficient test procedures and hardware can economically be obtained for memories. Mixing memory with general logic requires (relatively expensive) simulation of memory in test software. Additionally, such software does not generally determine if all memory locations have been tested, unless a (costly) gate equivalent simulation is performed for the entire memory.

Data flow should be partitioned so the module outputs are those needed for tactical purposes.

Circuits containing indistinguishable faults should be partitioned on the same replaceable unit where possible. Test verification software can indicate such faults.

Although redundant units can be compared to each other with the same test procedures, redundancy within a unit increases difficulty of detecting and locating single faults. Redundant logic within a module generally requires additional logic or test access to test each redundant path. Presence of an undetectable fault on a redundant line can invalidate a path sensitized for detection of other faults.

USE OF SYNCHRONOUS (CLOCKED) LOGIC

Use of synchronous logic whenever possible simplifies the development of test sequences. Because of individual delay ranges of portions of the logic on the module, difficulties are encountered when asynchronous logic is used. In synchronous circuits, clock pulses can prevent signals from flowing around loops. Also unknown values are less apt to be generated. Asynchronous logic should be allowed only where technically justified.

SCT logic provides inputs synchronously, with a fixed number of bits provided to the functional logic in parallel. Response patterns are also generated on a synchronous basis. Any inputs or outputs occuring asynchronously will require additional test circuitry.

FEEDBACK

Feedback loops present problems to effective test. Difficulty in initialization, and the need in many cases to propagate test data around the loop can result in long sequences in providing meaningful test.

Feedback should be used sparingly and with caution. If all feedback loops can be logically opened during test, the test generation problem is greatly simplified.

CIRCUITS FOR HIGHLY TESTABLE LOGIC

All circuits with memory should provide set or clear inputs. Both are desirable. The additional clear can be used for test purposes, while not interfering with use of the other clear input for normal system purposes. Dual clock inputs to counter or shift register circuits perform a similar purpose. Ready test access to a clock within a long counter can greatly reduce test sequence length.

Circuits with undefined output states, such as the RS flip-flop, should be avoided.

Combinational circuits such as multiplexers and decoders should have over-riding enable or strobe inputs. These control lines allow immediate entry of known values for test. Generally, all clocked circuits should trigger on the same edge.

MICROCOMPUTER PERIPHERAL LOGIC

SCT may be enhanced through the use of microcomputers. Using this method a module which has successfully completed self-test may, in turn, be used for pattern generation, test control and response encoding to allow SCT principles to be applied to computer peripheral logic. A corresponding saving of SCT hardware results from use of the microcomputer for this purpose. A peripheral may be considered to be any digital device subject to computer control for SCT purposes. Certain analog devices may be included as well, with between limits comparison made by computer software after analog/digital conversion.

The peripheral device, in general, will execute a computer or peripheral SCT control test sequence, and transmit to the computer response data from each module for each test pattern. Isolation to a single module is obtained by computer encoding of all response data from each module and comparison to the expected value. The

computer can determine the specific faulty module by software examining a simple table giving peripheral module logic flow order. Feedback between peripheral modules may be handled as in normal SCT use.[4]

PARALLEL CODE CHECKER ON DATA BUS

One parallel code checker, placed on the data bus, can perform encoding of all peripheral module responses in place of computer software.

First, data may be encoded by the parallel checker as it is placed on the bus by the module under test. This will require a Parallel Input (PIN) instruction to read (i.e., encode) each 16 bits of data from a module. Before new response data is encoded for a module, the encoded result of all previous response patterns must be placed in the code checker register. This would require two I/O instructions, a pulse output (IOCP) instruction to clear the register followed by a Parallel Output (POUT) instruction to load the code checker register with the previous code word from computer memory.

The second approach reads a number of values to be encoded from each module into temporary storage in computer memory. Assume, for example, 16 words (8-32 bit responses) are stored in computer memory before any encoding is performed. The data bus code checker is made ready to receive code values. This would require, as for the first case, two instructions. Then 16 successive "POUT" instructions addressing the code checker cause the 16 responses to be encoded. The encoded value, including encoding of the last 16 response words, is then read back to the computer.

The hardware of Figure 2-4 totals 11 standard "ICs", consisting of four exclusive "OR" Packages, two "hex registers", three "NAND" gate packages, and two tristate buffer packages. This hardware should be placed on the first external module in logic flow order, so that any faults in the data bus code checker will be correctly isolated to the module containing the checker.[4]

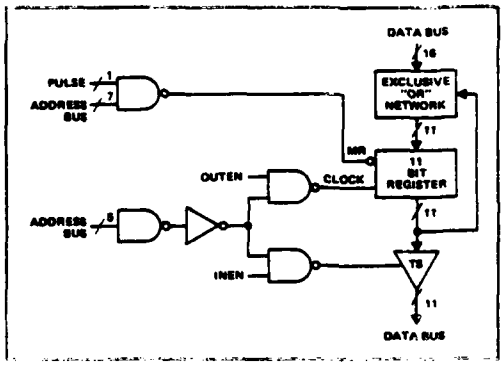


FIGURE 2-4. Parallel Cyclic Code Checker on Data Bus

SECTION III

TEST AND TEST CONTROL LOGIC

LOGIC FLOW

The control logic provides clocking as well as control sequences for critical logic paths that would otherwise be difficult to test. In a typical case the control logic would control multiplexers or I/O gates in the signal flow path. The control logic can also block feedback lines between modules so that isolation to a module can be achieved. At the conclusion of a test the control logic interrogates each pattern checker and (optionally) does any necessary decoding to locate the module which is at fault.

To implement the test functions above, the group of modules under test should include the following logic sections (as shown in Figure 3-1: (1) master timing control, (2) pseudorandom pattern generator, (3) sequence counter/decoder, (4) test complete and fault isolation logic, and (5) the pattern checker (the pattern checker is not really considered to be test control logic, but is part of the SCT functions).

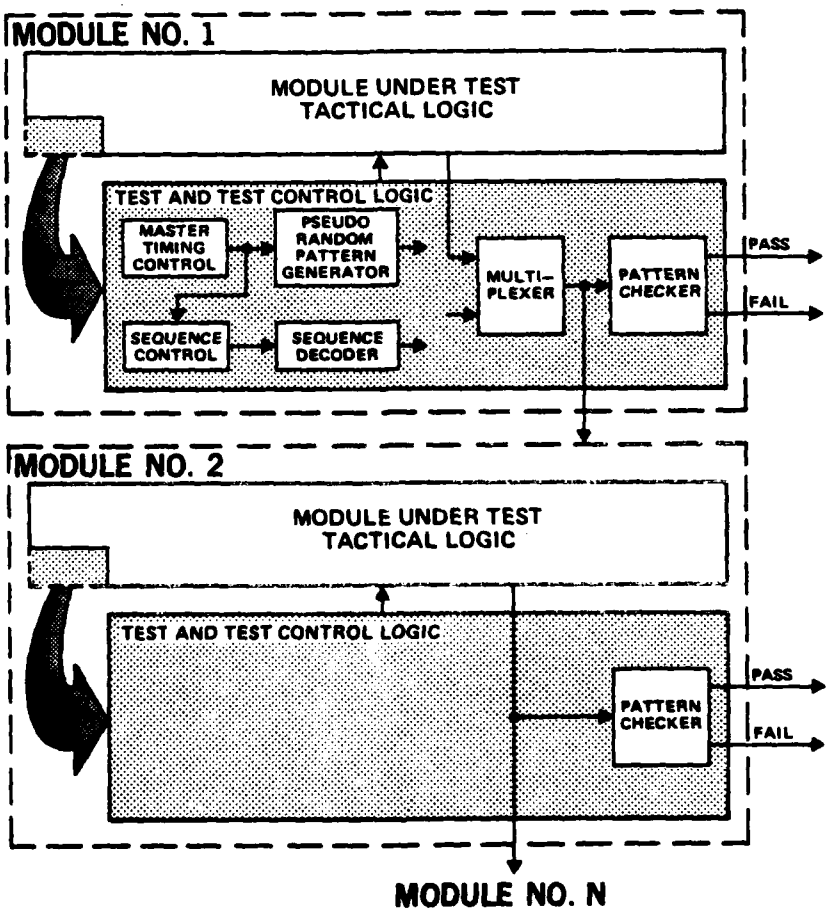


FIGURE 3-1. Example of Self Contained Test (SCT) Logic

The master timing control is derived from a master oscillator chosen to run at a frequency compatible with the logic being tested. Signals derived from the master timing control include the system clock, master clock pulse (MCP) and the multiplexer (MUX) select and pattern checker clock lines. The master timing thus selects one input of a 1 of 3 multiplexer at each pattern checker and clocks the data into the checker once for each set of inputs selected. 1 of 3 multiplexing is used with the availability of the 3 X 2 and-or-invert VLSI circuit in all VLSI tactical designs.

The pseudorandom pattern generator is a shift register in which certain stages are fed back to the input by way of an exclusive "OR" gate. Equations can be written to observe the shift register function in terms of modulo two addition. If the equation for the shift register feedback sequence is chosen so that the polynomial describing the register is primitive as described by Peterson (5), then the number of pseudorandom numbers that can be generated before repetition is given by $N = 2^K-1$ where N is the number of pseudorandom numbers and K is the number of flip-flops in the feedback shift register.

The sequence counter/decoder (test complete and fault isolation logic) is used to generate the control signals for the various modules and determine which portion of the test sequence is being executed. After initialization, the counter counts the master clock pulses. Decoding the binary count produces the necessary timing signals. There are two main phases to the test control sequence: during the first pulse phase all feedback lines between modules are blocked; during the second phase the feedback lines are not blocked and data flow is allowed through feedback paths. Blocking the feedback paths allows fault isolation to a single module by preventing data flow (which may be erroneous) from passing between modules in the feedback direction.

In general, control signals need not be generated at a precise time. Rather, control signals may be generated at time intervals which are sufficiently long for desired test and which can be decoded with a minimum of hardware. The control signals may also be shared by several modules, saving hardware. Control signals may be tailored to fit a particular module, but may be used for different purposes on another module.

The pattern checker is also a shift register with feedback paths similar to the pseudorandom pattern generator. The pattern checker is implemented on each module and is used exclusively by that module to verify the integrity of its logic. Pattern checkers are not conveniently shared by several modules due to the fact that an error in pattern checker logic could not be distinguished from a module logic error. Sharing a pattern checker, therefore, prevents complete isolation to the module level. Additionally, each module will generate its own checker pass/fail output telling whether the checker detected an error in the sequence of data flowing through the module.

MASTER TIMING CONTROL

A detailed logic diagram of an example of a master timing control is shown in Figure 3-2. Figure 3-3 shows the timing waveforms associated with this circuit. In

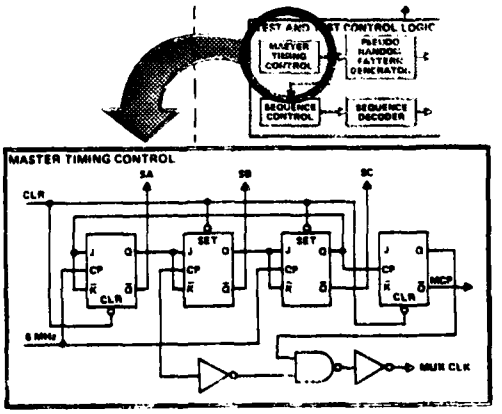


FIGURE 3-2. Example of Master Timing Control Logic

this configuration, the MCP frequency is equal to that of a 6 MHz signal divided by six. A nominal MCP frequency of 1 MHz could be as high as 4 MHz for operation in this system. This limitation would be imposed by the operating frequency of the shift registers making up the hypothetical module's memory. The 4 MHz limitation also restricts the fastest time in which tests can be completed.

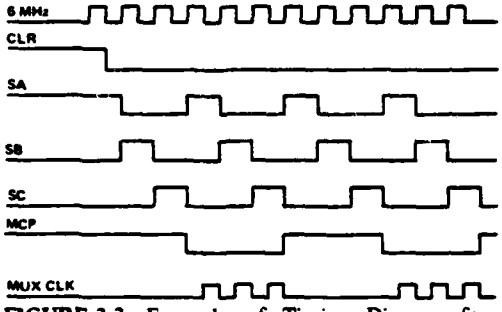


FIGURE 3-3. Example of Timing Diagram for Master Timing Control

The select lines (labelled SA, SB, and SC) are used by the pattern checker multiplexers on all modules. The select lines enable pattern checker inputs to be selected by 1 of 3 multiplexers formed by "3-2" and-or-invert VLSI circuits. To accomplish this, the first flip-flop of the recirculating shift register is initially cleared while the other two flip-flops are set. Thus only one Si signal is set, enabling one and-or-invert (AOI) input at each multiplexer clock.

The MUX CLK signal is the same as 6 MHz when MCP is low and is at logical zero when MCP is high. This is due to the fact that the logic used throughout the group of modules utilizes sequential circuitry that responds to a rising edge of a clock signal. If the pattern checker were clocked while the MCP was high, no new data would be received.

The master clock pulse is used as a system clock by all modules. However, the clock going to the sequence counter/decoder as shown in Figure 3-1 and to the pseudorandom pattern generator is inverted from that going to all other points of the module. This allows the sequence counter/decoder and the pseudorandom pattern generator to change states on the falling edge of MCP while the other parts of the module respond to the rising edge of MCP. The system then does not have to cope with changing (test) control and (pseudorandom) data lines at a time when it is responding to a clock pulse.

SOURCE OF INPUT PATTERNS

To allow testing independent of other groups of modules, each group of modules contains its own input pattern source. Each input pattern bit may fan out to several module inputs, reducing the number of pattern generator bits required. Multiplexers are used to insert the patterns into the desired data paths. Generally, such multiplexing is only required on prime module groups' inputs (i.e., incoming signals from other groups of modules) or key control signals.

Input patterns may be stored in a Read-Only Memory (ROM). ROM storage allows use of any desired set of predefined test input patterns. Industry studies have shown that a set of random patterns will often exercise logic nearly as well as more intelligently generated patterns. Such patterns are particularly applicable with SCT since test points are internal to the module and are not subject to pin limitations. (Certain signals, such as clock, mode, tactical and SCT enable-disable inputs should optimally control the group of modules for test and should not be established on a pseudorandom basis.) Improved test access increases the probability that random patterns will propagate a fault to an observable point. Thus, hardware often can be saved by using a pseudorandom pattern generator (of fixed sequence) instead of a ROM.

The test pattern source and other test

control and multiplex logic is not hard core, as the module on which they are located will have its own pattern checker. A separate timer may indicate failure to complete testing due to a control fault.

USE OF ROMS IN A PATTERN CHECKER

A ROM would store an expected response for comparison with module outputs at each pattern. The input patterns to the module are stored in a ROM contained in that module, in a ROM that acts as a source for all modules, or as part of a computer microinstruction. The obvious advantage to storing the test patterns for a module is that both initialization and fault isolation are made easier and the number of I/Os required for SCT are reduced.

The control checker design will most often be of the cyclic code type due to its straightforward implementation, high error detection, moderate hardware, flexibility in design, and the possible ability to use control ROM data to simplify code comparison. Codes with generator polynomials of the form $1+X^n$ require less hardware if commercial Cyclic Redundancy Code (CRC) chips are not used.

If a constant ROM is used in tactical hardware, it may be placed on computer data paths, this capability may be used to test the ROM and will also be a source of input test data.

Certain module signals, such as key control inputs for tactical and SCT logic, should be determined to optimally control the module for test. ROM use inherently provides this control. If pseudorandom pattern generation is used, times (patterns) at which a change in the value of a control signal is desired would be decoded from the pattern counter. The decoded pattern number may allow a flip-flop representing the control signal state to toggle at the next clock time. A circuit which would provide this control is shown in Figure 3-4. Alternately, a small ROM could store these key control signals. The pattern counter would, therefore, feed the address inputs of this ROM.

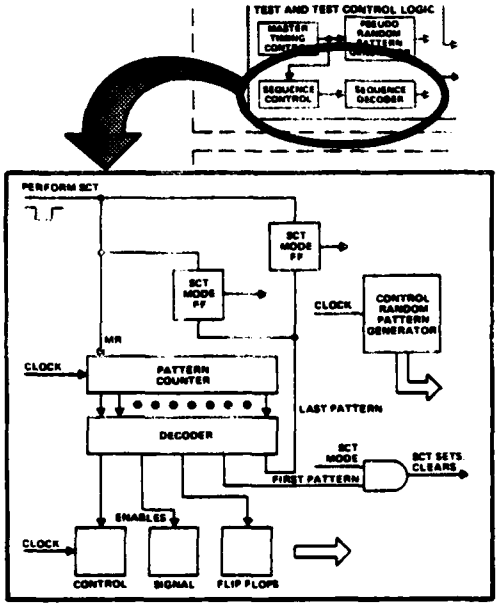


FIGURE 3-4. SCT Control Logic

THE PSEUDORANDOM PATTERN GENERATOR

The pseudorandom pattern generator shown in Figure 3-5 is a 12-bit feedback shift register. The particular generator used may be represented by the primitive generator polynomial $G(X) = 1 + X + X^4 + X^6 + X^{12}$. This polynomial was chosen because all other primitive twelfth degree polynomials require five or more feedback lines for their implementation. The polynomial can be efficiently implemented using two HEX D register circuits and two full adders with an extra $\overline{JK}$ flip-flop to insure the generator does not start in an all zero state.

Each MCP clock produces a 12-bit pseudorandom number. The 12-bit output changes on the falling edge of MCP and remains stable until the next falling edge.

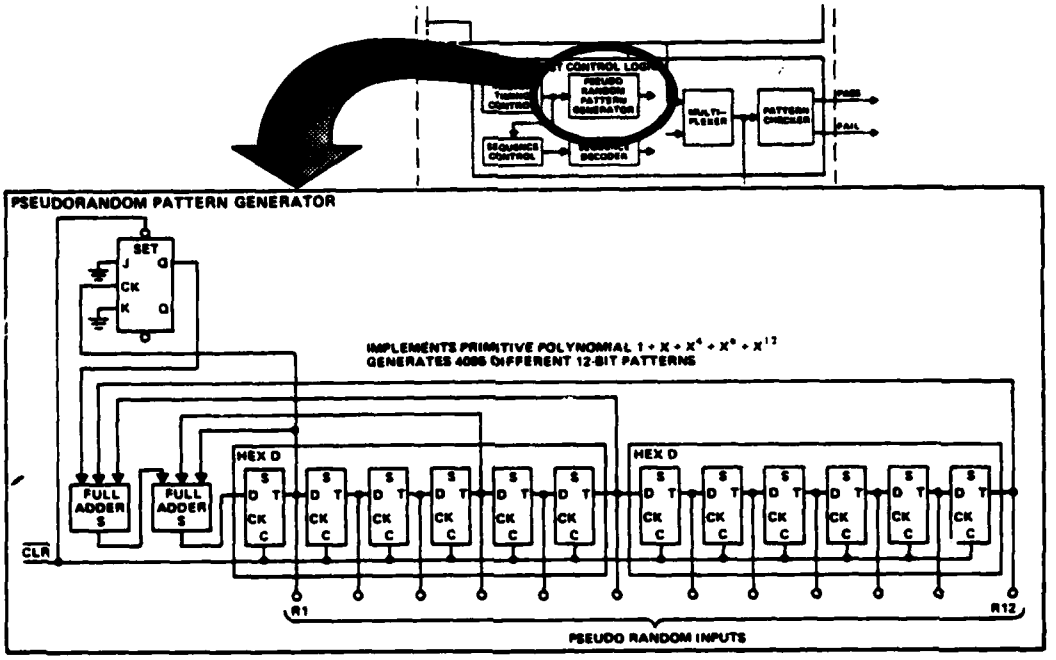


FIGURE 3-5. Pseudorandom Pattern Generator

Note that the pseudorandom pattern generator is a deterministic machine characterized by its initial state and its feedback equation. Therefore, its outputs are not truly random. However, if a single line is chosen, the pulses appearing on that line have the characteristics of pulses appearing in random time slots. Two or more lines looked at simultaneously will not necessarily have the characteristics of two or more random lines which are independent of each other. This characteristic may become significant when several random inputs interact. Although this type of generator cannot be designed to produce a preselected set of patterns, the patterns are known once the design is given.

THE COUNTER/DECODER

The purpose of the counter in the module under test control logic is to count clock pulses after test initiation to provide test event timing. The decoder monitors the counter outputs (number of clock pulses into the test sequence) and produces signals defining certain test periods. Some decoder outputs may be of long duration (up to half the total test length) while others may be only one clock pulse long. The purpose of the decoder is to produce timing signals suitable for test control on all modules.

An example of an 11-bit counter is shown in Figure 3-6. As it is used in the module under test the counter requires only 9 bits, but two extra bits are carried to provide a longer test if necessary. The MCP coming into the counter is inverted so that the counter changes states on the falling edge of MCP. The counter is synchronous while the clear is asynchronous and independent of the clock.

The counter may be cleared when CLR goes to a logical "one" or when CRESET goes to a logical "one". The CLR function

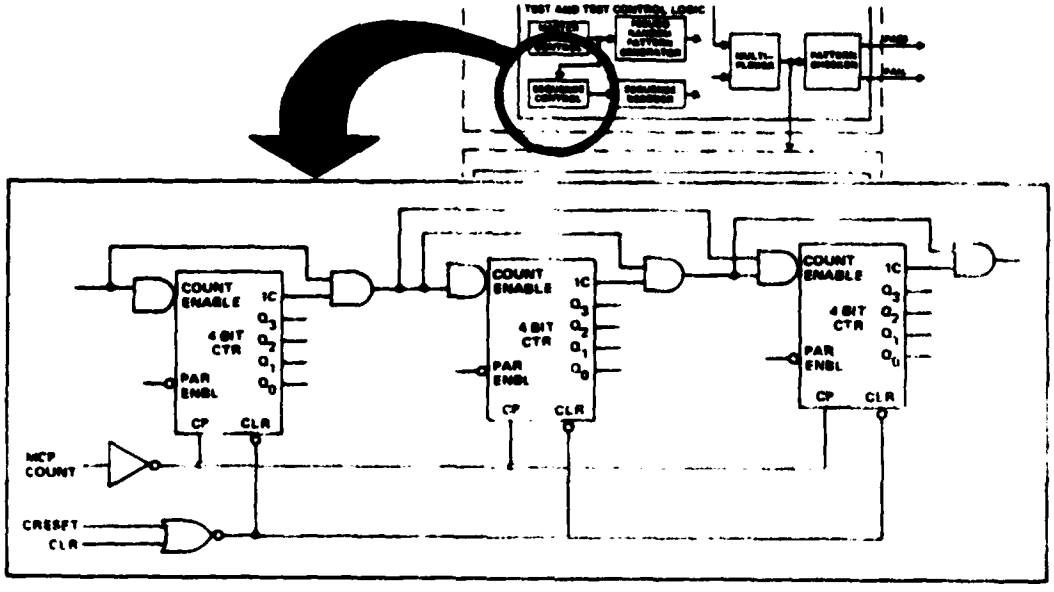


FIGURE 3-6. An Example of an 11-Bit Counter Used for Control Timing

is used to insure that the counter starts in the all-zero state. *The CRESET function* is used to reset the counter midway through the test procedure. This allows the same decoding logic to be used a second time to get the same control signals for the second (feedback enable) part of the test as for the first. If the counter were not reset, additional decoding logic would be required. Control signals are repeated during the second part of the test when the module logic is reconfigured to permit inter-module feedback.

PATTERN CHECKER/ COMPARATOR

The pattern checker used for SCT would consist of a hardwired comparator and a parallel cyclic code checker (generator). *Parallel cyclic code major circuit requirement is for* exclusive OR gating to provide in parallel the exclusive OR operation which otherwise would be performed serially. Multiplexing into the parallel inputs may be used to reduce the parallel code hardware needed. One level of AOI gates, used for input multiplexing will efficiently reduce the number of parallel inputs to that which may be handled with a moderate number of cascaded inputs. Using such multiplexing, 3 clock cycles are then needed to code one output pattern with each clock cycle coding 1/3 of the checked outputs in parallel.

A typical parallel cyclic code generator which can be used for SCT pattern checkers is shown in Figure 3-7. The generator polynomial $1 + X^2 + X^{11}$ requires 2 exclusive OR functions per cascaded input. The cascade construction allows nearly any number of inputs (limited by cumulative gate delay) to feed the 11-bit register in parallel. With 14 cascaded inputs (to check 42 output bits with 1 of 3 input multiplexing) 28 exclusive OR functions are needed. Each exclusive OR may be constructed with 4 NAND gates (6 interconnects) or an AOI and 2 NAND gates (6

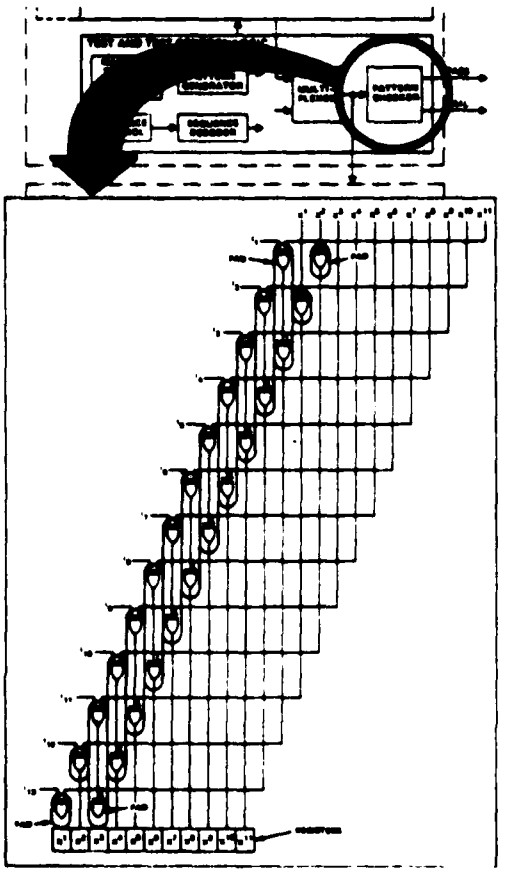


FIGURE 3-7. Hardware Implementation of Typical Code Generator

interconnects). If both polarities of the inputs are available, the exclusive OR may be implemented with a single AOI and 4 interconnects.

Due to the structure of the exclusive OR tree (Figure 3-7), a full adder circuit may be used to implement two exclusive OR functions in most cases. The sum output will provide the function (A ⊕ B) ⊕ C. Exclusive OR's needed to cascade 14 inputs may be constructed with 16 full adders, generally with one full adder (and 3 interconnects) per two exclusive OR functions. Only if both polarities of inputs were available for an AOI implementation would less power be used by a gate implementation. In all cases more interconnects would be required.

The full adder circuit is already desirable for VLSI post processor tactical logic implementation as it will allow power saving in replacing the 4-bit full adder circuits used in tactical logic.

Cyclic codes are well suited for detecting faults in large data words that are available serially one bit at a time to the code generator. The detection probability is nearly independent of the number of data bits.

The response patterns are fed into a multiplexer and then to the cyclic code generator one bit at a time. The final cyclic code bit number stored in the code generator can then be compared to a hard-wired constant. A mismatch implies the module is faulty. (See Figure 3-8.)

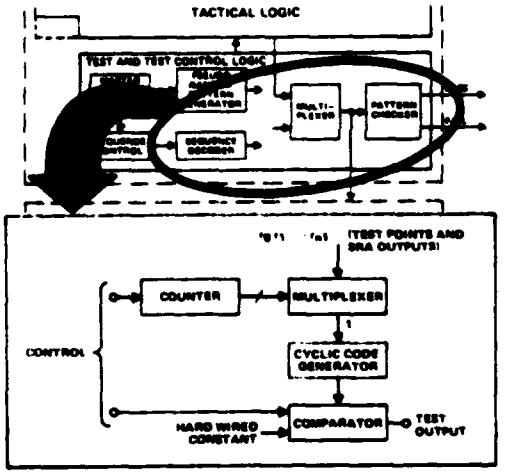


FIGURE 3-8. Test Comparator Logic

COMPARATOR LOCATION

Comparators may be provided on each module using the circuit types available on that module, or all pattern checker results may be compared by a single central comparator. In either case, a fault in a com-

parator gate may mask a tactical logic fault if the fault produces a "stuck-at-good" condition.

LSI 3 X 2 AOI circuits may be used in a single central comparator to multiplex in the sets of values to be compared. Three AOI's can multiplex in one bit of each of the seven final pattern checker code words to be compared. The true or inverted value of each checker output bit is sent to the central checker to allow comparison against all "one's" constant. Comparison of the input word to an all "one's" word can then be made with NAND gates. The 11 bits of each comparison word are input to 2-6 input NANDS, each followed by an inverter to produce the AND of the input bits. A final 2 input NAND will thus be zero only if all comparator inputs are one. Alternatively, the 6 input NAND outputs, which are zero only if no fault exists, may be input to an AOI as shown in Figure 3-9. The AOI output will be one only if the comparison word is all ones. These values may be stored in a shift register. A total of 36 AOIs and 23 NANDs are required for multiplexer select line buffering, multiplexing and the compare network.

Using the design in Figure 3-9, a single comparator on a VLSI can be constructed with 2-6 input NANDs and an AOI (or 3 additional NANDs). For 6 comparators, 12 NANDs and 6 AOIs are required.

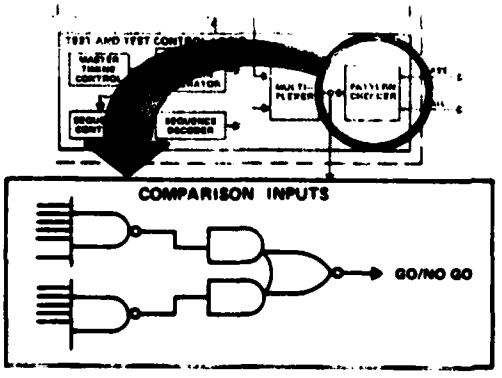


FIGURE 3-9. Eleven-Bit Comparator using LSI NAND and AOI Circuits

PARALLEL CYCLIC CODE IMPLEMENTATION

CRC generators are used in data transmission systems and data storage systems to detect errors in received data. They are capable of detecting a very high percentage of errors in data records of any length. The main reasons for parallel CRC generator construction are: (1) speed-it takes far less time to process data in parallel than in serial form; and (2) in most data processing systems where data move in parallel, clock pulses which perform conventional serial processing to obtain the CRC word may not be available.[6]

By cascading 16 generator parts the combinational circuit of Figure 3-10 is obtained which processes the 16 bits currently in the CRC register (CO-C15)

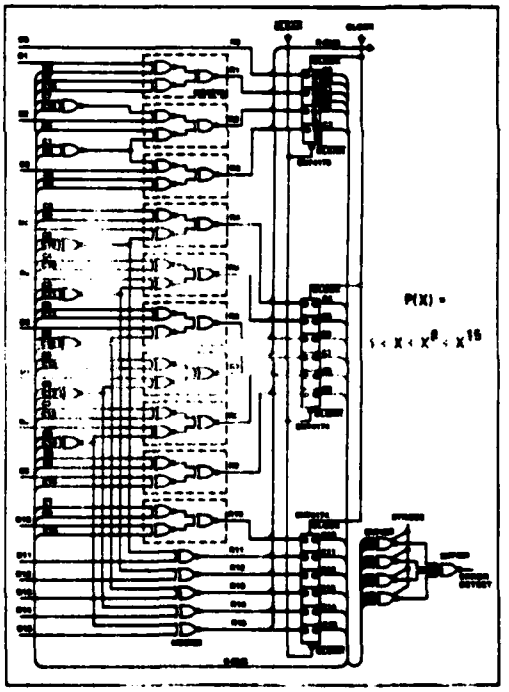


FIGURE 3-10. Example of High-Speed Parallel CRC Generator

and the 16 incoming data bits (DO-D15) and generates the remainder polynomial (RO-R15). This is equivalent to dividing a 32-bit word by a 16-bit word, Mod 2, and obtaining the 16-bit remainder. RO-R15 are the inputs to the same register (D type flip-flops). This circuit requires 12 quad exclusive OR gate packs and one 16-bit register (three ICs).

It is possible to reduce the IC count and delay in the circuit. Let us set up the equations and find the R variables expressed by the C and D variables. In doing so, it is easiest to find IO-I15 first. (IO-I15 are intermediate variables that determine whether or not a particular stage should perform the Mod-2 addition.)

```
I15 = C15
I14 = C14
I13 = C13
I12 = C12
I11 = C11
I10 = C10
I 9 = C 9
I 8 = C 8 ⊕ I15 = C 8 ⊕ C15
I 7 = C 7 ⊕ I14 = C 7 ⊕ C14
I 6 = C 6 ⊕ I13 = C 6 ⊕ C13
I 5 = C 5 ⊕ I12 = C 5 ⊕ C12
I 4 = C 4 ⊕ I11 = C 4 ⊕ C11
I 3 = C 3 ⊕ I10 = C 3 ⊕ C10
I 2 = C 2 ⊕ I 9 = C 2 ⊕ C 9
I 1 = C 1 ⊕ I15 ⊕ I 8 = C 1 ⊕ C 8
I 0 = C 0 ⊕ I15 ⊕ I14 ⊕ I7 = C 0 ⊕ C 7 ⊕ C15
```

In evaluating I 1 and I 0, note that (C 1 ⊕ C 15 ⊕ C 15) = C 1. Also note that the associative, distributive and commutative laws are valid for Mod 2 addition. Now we can evaluate RO - R15.

```
R15 = D15 ⊕ I14 ⊕ I13 ⊕ I 6 = D15 ⊕ C14 ⊕ C 6
R14 = D14 ⊕ I13 ⊕ I12 ⊕ I 5 = D14 ⊕ C13 ⊕ C 5
R13 = D13 ⊕ I12 ⊕ I11 ⊕ I 4 = D13 ⊕ C12 ⊕ C 4
R12 = D12 ⊕ I11 ⊕ I10 ⊕ I 3 = D12 ⊕ C11 ⊕ C 3
R11 = D11 ⊕ I10 ⊕ I 9 ⊕ I 2 = D11 ⊕ C10 ⊕ C 2
R10 = D10 ⊕ I 9 ⊕ I 8 ⊕ I 1 = D10 ⊕ C10 ⊕ C 8 ⊕ C 1
R 9 = D 9 ⊕ I 8 ⊕ I 7 ⊕ I 0 = D 9 ⊕ C14 ⊕ C 9 ⊕ C 8 ⊕ C 0
R 8 = D 8 ⊕ I 7 ⊕ I 6 = D 8 ⊕ C14 ⊕ C13 ⊕ C 7 ⊕ C 6
R 7 = D 7 ⊕ I 6 ⊕ I 5 = D 7 ⊕ C13 ⊕ C12 ⊕ C 6 ⊕ C 5
R 6 = D 6 ⊕ I 5 ⊕ I 4 = D 6 ⊕ C12 ⊕ C11 ⊕ C 5 ⊕ C 4
R 5 = D 5 ⊕ I 4 ⊕ I 3 = D 5 ⊕ C11 ⊕ C10 ⊕ C 4 ⊕ C 3
R 4 = D 4 ⊕ I 3 ⊕ I 2 = D 4 ⊕ C10 ⊕ C 9 ⊕ C 3 ⊕ C 2
R 3 = D 3 ⊕ I 2 ⊕ I 1 = D 3 ⊕ C 9 ⊕ C 8 ⊕ C 2 ⊕ C 1
R 2 = D 2 ⊕ I 1 ⊕ I 0 = D 2 ⊕ C 8 ⊕ C 8 ⊕ C 7 ⊕ C 1 ⊕ C 0
R 1 = D 1 ⊕ I 0 = D 1 ⊕ C15 ⊕ C 7 ⊕ C 0
R 0 = D 0
```

Note that at least two terms in each equation cancel each other. This results from careful selection of the generator polynomial P (X).

The method of reducing circuitry by finding duplicate terms in the equations can apply to any generator polynomial P(X). Some polynomials yield circuits more suitable for easy implementation than others, but these tend to belong to lower exponents. Because the search for an optimum generator polynomial was tedious and time consuming, a computer program was written to evaluate polynomials and print out those which looked promising. Final evaluation of polynomials selected was done by hand.

A parallel 16-bit cyclic code generator has been implemented using parity generators and Exclusive-OR gates. The generator requires only one clock pulse to process 16 bits of information, and has a cycle time of 105 ns.

SECTION IV

SCT EFFECTIVENESS

Knowing the effectiveness of the module test response coding (P_c) and the probability of module faults being propagated to code checker inputs ($P_O(X)$), fault detection and isolation may be computed for feedforward logic. If each module feeds only one module following in logic flow order (as in Figure 1-1), the probability of fault detection and isolation to i or fewer modules is given by

$$P_{Di} = P_c \sum_{j=0}^{i-1} (1 - P_c)j \prod_{K=0}^{j} P_O(nE^K) \quad (1)$$

$$P_{Ii} = P_c \sum_{j=0}^{i-1} (1 - P_c)j$$

Where

P_{Di} = the probability that a random fault in the k level module will be detected by SCT at least once, looking at all module pass/fail outputs from the k level module to the k + j level module.

P_{Ii} = The probability that detected faults will be isolated to j or fewer levels of modules

P_c = the probability that an erroneous test response from a module will be detected by SCT coding P_c can be determined from the properties of the code used

$P_O(x)$ = The probability that, with X group of modules test patterns, a random fault in a module will produce an erroneous test response from that module. Where the full set of "N" input patterns is referred to, this value becomes $P_O(n)$. This number can be determined by test simulation software for each module.

E = the probability that an erroneous j - 1 level module test response will produce an erroneous test response in an level module.

The group of modules as shown in Figure 1-1 consists of three levels: Module 1 is the first level, module 2 the second, and module 3 the third. A module 1 fault can be detected on that module, and in the two module levels fed by module 1. The detection probability of a module 1 fault is thus P_{D3} since it can be detected in the three levels. Module 2 faults may be detected by code checkers on module 2 and module 3. Module 2 fault detection probability is thus P_{D2}.

For a single module, fault detection (P_{D1}) is given by $P_c P_O(n)$. Isolation to the exact faulty module (P_{I1}) is given by P_c. Practical coding techniques allow P_c of over 0.99. Thus nearly all faults propagated to test points or module outputs will be detected by SCT, and isolation to a single faulty module achieved for nearly all detected faults. Figure 4-1 shows values of P_c and $P_O(n)$ which will provide desired

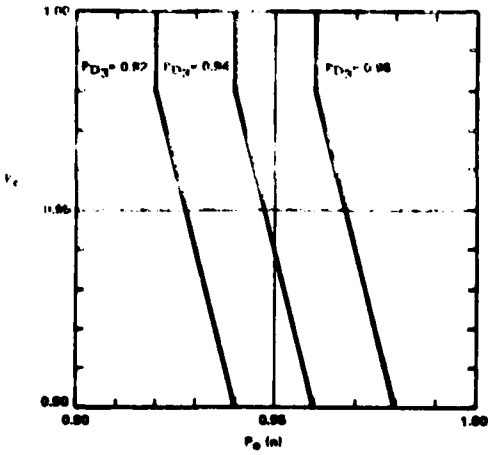


FIGURE 4-1. Graph Showing Module Test Completeness

high percentages of module test completeness. It exhibits a near linear relationship for P_C and P_O (n) values graphed. As P_C increases, the required P_O (n) for a given P_{Di} decreases. However, P_{Di} is much more sensitive to changes in P_O (n) than changes in P_C. Figure 4-2 illustrates the near linear relationship between P_{Di} and P_O (n) for a fixed P_C. Curves are drawn for expected values of P_C.

Figure 4-3 represents SCT isolation capabilities. Considering isolation of detected faults to i or fewer levels, P_{Ii} improved isolation results as SCT fault detection P_C increases. For constant P_C, isolation to 2 or 3 levels is approximately the same. Nearly all faults are isolated to 2 levels for expected P_C.

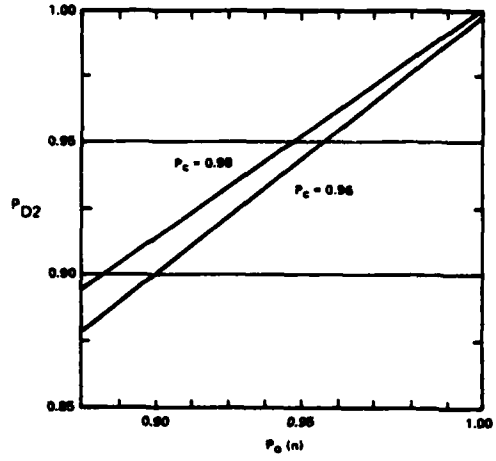


FIGURE 4-2. Graph Showing Linear Relationship Between P_{Di} and P_O (n) for Fixed P_C

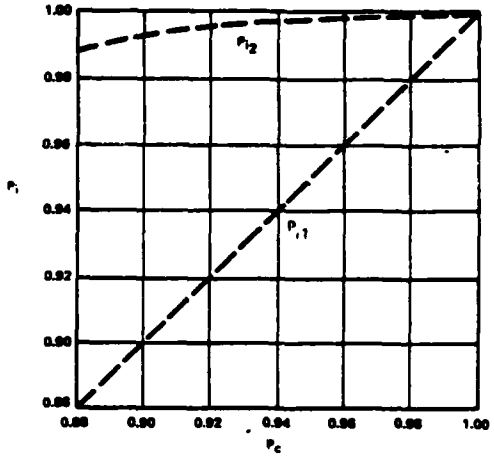


FIGURE 4-3. Graph Showing SCT Isolation Capabilities

SECTION V

CONCLUSIONS

Self-Contained-Test provides economic and highly complete test and diagnosis in complex digital modules. Application of SCT is enhanced by the availability of low-cost circuits. With the SCT concept providing test control and a single pass/fail test result from each module, ATE can be greatly simplified with significant cost and space savings. Module isolation is performed by SCT using simple pattern comparison. SCT design is direct and systematic. Fault detection and isolation in complex groups of modules thus becomes manageable and may be approached in a straightforward manner. Important cost savings are also realized over conventional techniques addressing module test.

While allowing straightforward tests in complex groups of modules with significant cost savings, the SCT concept also realizes the following advantages:

(1) Test access is greatly simplified by placing the test function integral to the prime equipment. Additionally, few module I/Os are required for SCT.

(2) The prime equipment designers, who know the equipment best, are responsible for design of the test logic. They can also be highly motivated to provide testable prime equipment design through computer-aided analysis of their designs' testability. Partitioning for test can be more readily accomplished with the test functions defined as an integral part of the overall logic. Different manufacturers can more easily provide second - source capability since the module test interface can be identical even though logic implementations differ.

(3) Fault diagnosis programs for automatic testers can be substantially reduced in complexity and, in some areas, eliminated entirely. The cost of test software, although mostly non-recurring, is a significant element in the cost of current automatic test systems.

(4) Incorporation of the test function into the prime equipment should simplify the test process so that maintenance and test resources can be utilized with greater flexibility. Personnel skill level requirements can be relaxed, external test equipment can be more nearly standardized, and the prime equipment can be made compatible with several testers, for example.

(5) Module status information obtained with SCT may be used to enhance operational effectiveness through continuous monitoring and degraded mode assessment. For example, since the Self-Contained-Test concept allows test sequence execution at up to the normal system clock rate, a 1 MHz clock can provide the comparison of 10,000 test patterns in 10ms. Isolation information from tests performed in an operational (e.g. airborne) environment can allow effective maintenance action to be taken even for faults which are not duplicated in the shop.

(6) In highly reliable systems which require continuously powered redundant subsystems with "voted" results to detect a failure and allow switching, SCT presents an interesting alternative. SCT could allow only one unit at a time to be under power and operating with a "fail" SCT output providing the signal to switch to a standby unit. As a result, less power and fewer units would be necessary for the same level of performance and reliability as only one

rather than three units must operate to assure no failed results.

(7) SCT logic requirements are moderate, and no special circuit types are needed. SCT design is direct and will not add delays to groups of modules feed-forward data paths.

(8) Isolation to groups of modules and individual modules is extremely high, and fault detection is high. Such results will be obtained for any fault type (including multiple faults) which will produce erroneous module output states.

SECTION VI

GLOSSARY OF TERMS

The following is a collection of terms related to testability with their definitions. The source of each definition is given in parentheses following the definition. The code used is:

MS1309B: MIL-STD-1309B

MS721B: MIL-STD-721B

ATG: Industry ATG Glossary, Report of Industry Ad Hoc ATE Project for the Navy, April 1977.

IEEE: IEEE Standard Dictionary of electrical and electronics terms, IEEE Std. 100-1972.

IEEE/FTC: Interim IEEE Technical Committee of Fault Tolerant Computing Dictionary of Terms.

P73-314: Vol. 2 AAFIS Concepts, Hughes.

(Blank): Author's definition.

1. Active Redundance: That redundance wherein all redundant items are operating simultaneously, rather than being switched on when needed. (IEEE)

2. Active Testing: Closed-loop testing.

3. Adapter: A device designed to provide a compatible connection between the unit under test and the test equipment.

4. Ambiguity Group: The group of maintenance replaceable units which may contain faults which result in the same fault signature; also the number of units in such a group.

5. Automatic Test Equipment (ATE): Equipment that is designed to conduct analysis of functional or static parameters to evaluate the degree of performance degradation and may be designed to perform fault isolation of unit malfunctions. The decision making, control, or evaluative functions are conducted with a *minimum* reliance upon human intervention. (MS 1309B)

6. BIT (Built-in Test): A test approach using BITE or self test hardware or software to test all or part of the UUT. (MS1309B)

7. BITE (Built-in Test Equipment): Any device which is part of the equipment or system and is used for the express purpose of testing that equipment or system. BITE is an identifiable unit of the equipment or system. (MS1309B)

8. BIT Slice: A logic partition in which certain bits of different logic functions (e.g., multiplexing, arithmetic logic and registers) are placed on each of several partitioned units. (P73-314)

9. Bridge Fault: A short fault.

10. Casualty: A manifestation of a failure at the system level or major subsystem level such that the system/subsystem is incapable of performing its principal function(s). A casualty is differentiated from a malfunction by the greater seriousness or persistence of its nature.

11. Catastrophic Fault, Analog: A fault in analog circuitry which causes a sudden change in operating characteristics and results in a complete lack of useful performance. (ATG)

12. Catastrophic Fault, Digital: A primary failure in digital circuitry which causes secondary failures.

13. Checkpoint: A place in a routine where a check, or recording of data for restart purposes, is performed. (IEEE)

14. Circuit Package: The smallest replaceable module of a system. Commonly a flat pack or dual-in-line SSI/MSI/LSI. (P73-314)

15. Closed-Loop Testing: Testing in which the input stimulus is controlled by the equipment output monitor. (MS1309B)

16. Component Internal Fault: A device or component fault which is not a pin fault.

17. Continuous Test: Test performed during operational conditions.

18. Confidence Test: A go/no go test.

19. Controllability: An attribute of equipment design which defines or describes the degree of test control which may be exercised at internal nodes of interest.

20. Critical Race: In digital logic, the concurrent change of two or more feedback lines which may result in any one of two or more stable states being entered.

21. Defect: A property of a device which is outside its specified limits in one or more parameters.

22. Delay Fault/Delay Failure: A failure in a digital device such that switching occurs to the proper level but does so outside of a specified time interval.

23. Dependent Fault/Dependent Failure: A fault which is caused by the failure of an associated item. (MS721B)

24. Design Fault: A design characteristic of either hardware or software which causes or materially contributes to equipment malfunction independent of the presence of hardware failures.

25. Design for Testability (DFT): A design process or characteristic thereof such that deliberate effort is expended to assure that a product may be thoroughly tested with minimum effort and that high confidence may be ascribed to test results.

26. Detectable Faults: Faults whose test results differ with those obtained from a good system for at least one possible input pattern. (P73-314)

27. Detection Percentage: The percentage of the total number of equipment faults which are detected by a given test procedure. (P73-314)

28. Diagnostic Accuracy: The percentage of the total number of faults under consideration that can be correctly diagnosed by the simulated faults. Diagnostic accuracy is a function of the following:

    a. Effectiveness of the diagnostic procedure; i.e., the coding used to represent test results, type dictionary used, etc.

    b. Completeness of the fault simulation process.

    c. System Organization: The system organization is affected by the "availability of easily accessible test points," "modularity of the logic design," and "optimality of the packaging techniques."

    d. Fault and model precision. (P73-314)

29. Diagnostic Test: A test designed to perform isolation of a fault to replaceable assemblies in the unit under test.

30. (Diagnostic) Test Results or (Diagnostic) Data: The responses or output patterns of a (Diagnostic) test. (P73-314)

31. Distinguishable Faults: Faults for which there exists at least one possible test which produces output patterns (test results) differing for the faults.

32. Dynamic Test: A test of one or more of the signal properties or characteristics of an equipment or any of its constituent items performed such that the parameters being observed are measured and assessed with respect to a specified time aperture or response. (ATG)

33. Early Failures: Also birth failures or burn-in failures. Failures during the early

period, beginning at some stated time and during which the failure rate of some items is decreasing rapidly. (IEEE) A large number of internal circuit failures probably occur in this period due to manufacturing imperfections or design errors.

34. Element: The logic unit analyzed by a test software program. May range from a gate to a group of gates, such as a portion of or an entire MSI component. (P73-314)

35. Error: Any discrepancy between a computed, observed or measured quantity and the true, specified or theoretically correct value or condition. (IEEE)

36. Equivalent Fault: A fault X is equivalent to a fault Y if, and only if, a system containing fault X has the same observable behavior as the same system containing fault Y. (IEEE/FTC)

37. Exact Match Fault Dictionary: A fault dictionary whose successful utilization is predicated exclusively upon existence of exact matches of observed fault signatures against predicted fault signatures enumerated in the dictionary.

38. External ATE: ATE which is physically separated from the unit under test when the UUT is in its operational environment.

39. Failure: A condition (electrical/mechanical) causing a device to be outside its specified limits in one or more parameters. (P73-314)

40. Failure Analysis: The logical, systematic examination of an item or its diagram(s) to identify and analyze the probability, causes and consequences to potential and real failures. (MS721B)

41. Failure Mode: A failure classification.

42. Fail Soft/Soft Failure: The toleration of the effects of a predetermined number of failures with only partial loss of functional capacity. (IEEE/FTC)

43. Failure Universe/Failure Population: The failures which correspond to a selected fault population. This is used as a basis for the design and evaluation of tests.

44. False Alarm: An indicated fault where no fault exists. (MS1309B)

45. False Alarm Rate: The frequency of occurrence of false alarms.

46. Fault: A physical condition that causes a device, component or element to fail to perform in a required manner; for example, a short-circuit or a broken wire. (IEEE)

47. Fault or Failure Mode: A manifestation of a physical defect or failure in a logic element which can cause incorrect or undesired network operations. (P73-314)

48. Fault Coverage: An attribute of a test or test procedure expressed as the percent of faults of the total fault population which that test or test procedure will detect.

49. Fault Detection: A process which discovers or is designed to discover the existence of faults; the act of discovering existence of a fault.

50. Fault Diagnosis: The process of detecting and isolating a fault. (P73-314)

51. Fault Dictionary: A list of elements where each element consists of a test and all the faults detected by that test. (IEEE/FTC) Often only the LRUs which contain the faults are listed.

52. Fault Dominance: A fault X dominates a fault Y if, and only if, every test for Y is a test for X, where X and Y are two faults which may occur in a system. (IEEE/FTC)

53. Fault Isolation: Where a fault is known to exist, a process which identifies or is designed to identify the location of that fault within a small number of replaceable units.

54. Fault Localization: Where a fault is known to exist, a process which identifies or is designed to identify the location of that fault within a general area of equipment. Fault localization may be less specific than fault isolation.

55. Fault Masking: A fault X masks a fault Y if no test for fault Y is a test for the faults X and Y occurring jointly in a system. (IEEE/FTC)

56. Fault Population: The totality of faults which may be incurred by a device.

57. Fault Prediction: A process used to predict that some component will be out of tolerance before the next scheduled maintenance period based upon the present measurement of component parameters.

58. Fault Resolution: A measure of the capability of a test process to perform failure isolation among replaceable units, generally expressed as (n) or fewer replaceable units XX% of the time (based upon the fault population).

59. Fault Signature: An output test vector resulting from the testing of a unit containing one or more faults.

60. Fault Simulation: A process which admits prediction or observation of system behavior in the presence of a specified fault without infliction of that fault upon that system. The process demands modeling of either the fault, the system, or both.

61. Fault Tolerance: The capacity of a computer, subsystem or program to withstand the effects of internal faults; the number of error-producing faults a computer, subsystem, or program can endure before normal functional capability is impaired. (IEEE/FTC)

62. Field Failures: In-service failures, characterized by an absence of burn-in failures.

63. Functional Partitioning: The physical or electrical separation of system elements along interfaces which define and isolate these elements on bases of function or purpose.

64. Functional Test: A test which is intended to exercise an identifiable function of a system. (IEEE/FTC) The function is tested independent of the hardware implementing the function.

65. Gate Equivalent Modeling: Representation of a complex logic device as a network of gates and other simpler logic devices.

66. Hard Core: That kernel of circuitry in a processor or system which must be functioning properly in order for that processor or system to successfully execute tests of other portions of itself.

67. Hard Core Failure: A failure in the hard core logic of a system which inhibits normal self-test of the system.

68. Hazard: In combinational logic, the possible transient changing of an output due to internal delay characteristics. A hazard is harmful if it affects the states of memory devices.

69. Impossible Detect: Failures which cannot be detected by any test. (ATG)

70. Independent Fault/Independent Failure: A fault which occurs without being related to the failure of associated items. (MS721B)

71. Initialize: (1) To establish an initial condition or starting state; for example, to set logic elements in a digital circuit or the contents of a storage location to a known state so that subsequent application of digital test patterns will drive the logic elements to another known state; and (2) To set counters, switches, and addresses to zero or other starting values at the beginning of or at prescribed points in a computer routine. (MS1309B)

72. Input Test Vector: A test pattern.

73. *Interence Testing: Off-line testing.*

74. Intermittent Fault: A temporary fault. (IEEE)

75. Inverted-Pyramid/Building-Block: Descriptive terms characterizing a test or test technique whereby the smallest possible portions of hardware are tested first in the test sequence and subsequent tests utilize previously verified hardware for execution.

76. Latent Fault Time: The extent or duration of time during which an existing fault is undetected; the elapsed time between fault occurrence and fault detection.

77. Line Replaceable Unit (LRU): A unit which is designated by the plan for maintenance to be removed upon failure from a larger entity (equipment, system) in the latter's operational environment. (MS1309B)

78. LSI: Large Scale Integration.

79. Maintainability: A characteristic of equipment design and installation which is expressed as the probability that an item will be retained in or restored to a specified condition within a given period of time, when the maintenance is performed in accordance with prescribed procedures and resources. (MS721B)

80. Maintenance Replaceable Unit: A LRU.

81. Malfunction: An error.

82. Mean Time Between Maintenance (MTBM): The mean time of the distribution of the time intervals between maintenance actions (either preventive, corrective, or both). (MS721B)

83. Mean Time To Isolate: The average time required to achieve fault isolation as measured from the time of fault detection to the time of fault isolation.

84. Mean Time To Localize: The average time required to achieve fault localization as measured from the time of fault detection to the time of fault localization.

85. Mean Time To Repair (MTTR): The total corrective maintenance time divided by the total number of corrective maintenance actions during a given period of time. (MS721B)

86. Mistake: A human action that produces an unintended result. (IEEE)

87. MSI: Medium Scale Integration.

88. Multiple Failure: A joint occurrence of two or more single failures. (IEEE)

89. Observability: An attribute of equipment design which defines or describes the extent to which signals of interest may be observed.

90. Off-Line Test: Test of a unit under test (UUT) with the unit removed from its normal operational environment. (ATG)

91. Off-Line Test Equipment: Equipment used to perform tests on a UUT with the unit removed from its normal operating environment. (ATG)

92. On-Line Test: Test of a UUT in its operational environment. (MS1309B)

93. On-Line Test Equipment: Equipment used to perform tests on a UUT while the unit is in its normal operating environment. (ATG)

94. Open Fault: A fault caused by an electrical connection between normally electrically connected points. (IEEE/FTC)

95. Output Test Vector: An ordered set of simultaneously observed output values.

96. Parametric Fault: A fault which causes some parameter for a device to have a value outside its specified range. (IEEE/FTC)

97. Parametric Test: The measurement of circuit characteristics to ascertain that they fall within specified tolerances. (ATG)

98. Passive Test: Non-active testing.

99. Pattern Sensitive Failure: A component failure, usually internal to the component, whose effect at the component's output pin(s) is dependent upon the input applied.

100. Pin Fault: A fault which is present at a single input or output pin of a component or module.

101. Primary Failure: An independent fault.

102. Random Fault/Random Failure: An intermittent fault whose occurrence is predictable only in a statistical sense.

103. Readiness: A state of being ready to successfully perform or being in the act of successfully performing a defined mission.

104. Readiness Test: A test specifically designed to determine whether an equipment or system is operationally suitable for a mission. (MS1309B)

105. Recovery: The continuation of system operation with error-free data after an error occurs. (IEEE/FTC)

106. Redundant Failure: A failure whose occurrence in a system does not terminate system ability to perform any required function. (IEEE/FTC)

107. Redundance, Redundancy: The introduction of auxiliary elements and components into a system to perform the same functions as other elements in the system for the purpose of improving reliability and safety. (IEEE) Also, the use of additonal components, programs, or repeated operations, not normally required by the system to execute its specified tasks to overcome the effects of failures. (IEEE/FTC)

108. Repeatability: A test characteristic such that repeated application of a given set of stimuli to a UUT yields identical results.

109. Response: The observable reaction of a device to stimulus.

110. Secondary Failure: One or more dependent faults.

111. Self Test: Built in test.

112. Solid Fault: A permanent fault.

113. Standby Redundance: That redundance wherein the alternative means performing the function is inoperative until needed and is switched in upon failure of the primary means of performing the function. (IEEE)

114. Static Test: A test in which measurement is made on a UUT after, and only after, these outputs have stabilized with respect to a given input stimulus. (ATG modified)

115. Stimulus: Any physical or electrical input applied to a device intended to produce a measurable response. (MS1309B)

116. Short Fault: A fault caused by an electrical connection between normally electrically separated points. (IEEE)

117. Specified Fault Population: A subset of the fault population which is used as the basis for defining the failure universe.

118. Stuck Fault/Stuck Failure: A failure in which a digital signal is permanently held in one of its binary states. (IEEE)

119. Symptom: An error which is the manifestation or evidence of a particular failure condition.

120. Test: A procedure or action taken to determine under real or simulated conditions the capabilities, limitations, characteristics, effectiveness, reliability, or suitability of a material, device, system or method. (MS1309B)

121. Testability: A design characteristic which allows the status (operable or inoperable) of a system or any of its subsystems to be confidently determined in a timely fashion.

122. Test Effectiveness: A measure which reflects the fault coverage and fault resolution provided by a test.

123. Test Generation: The process of designing tests or test stimuli.

124. Test Length: The number of tests in a test sequence.

125. Test Pattern: A simultaneous or parallel definition of all the inputs of a system. (IEEE/FTC)

126. Test Sequence: A specific order of related tests. (MS1309B)

127. Test Validation: Actions taken to determine if test responses for a fault-free UUT are in agreement with desired values.

128. Test Verification: Actions taken to assure that a test meets specifications of fault coverage and fault resolution.

129. Transient Failure: A failure induced by a momentary or temporary external factor such as input power fluctuation, excessive ambient temperature excursion, electromagnetic interference, or by factors internal to a system. A solid fault may cause a transient failure.

130. Wearout Failures: Failures during the period during which the failure rate of some items is rapidly increasing due to deterioration processes. (IEEE)

131. Well-Behaved Failure: A failure whose occurrence produces dependably consistent and predictable symptoms.

LIST OF ABBREVIATIONS

AOI: And or Invert
ATE: Automatic Test Equipment
ATP: All Test Pass
BIT: Built In Test
CKT: Circuit
CL: Clear
CLK: Clock
CLR: Clear
CRC: Cyclic Redundancy Code
CRESET: Clear Reset
D.C.: Demonstration Control
FAD: Full Adder
GFAD: Gated Full Adder
IC: Integrated Circuit
I/O: Input/Output
IOCP: Input Output Clear Pulse
LSI: Large Scale Integration
MCP: Master Clock Pulse
MD: Magnitude Detector
MHz: Mega-Hertz
MOD-2: Modulo Two
MR: Microinstructor Register
MSI: Medium Scale Integration
MUX: Multiplexer
PDI: Post Detection Integration
PIN: Parallel Input
POUT: Parallel Ouput
PPT: Post Processor Timing
ROM: Read Only Memory
SAO: Stuck At Zero
SA1: Stuck At One
SCT: Self Contained Test
SSI: Small Scale Integration
VLSI: Very Large Scale Integration

REFERENCES

1. Benowitz, N., Calhoun, D.F., Alderson, G.E., Bauer, J.E., Joeckel, C.T., "An Advanced Fault Isolation System for Digital Logic," *IEEE Trans Computers*, Vol. C-24, May 1975, pp. 489-497.

2. Alderson, G.E., Benowitz, N., Bork, R.H., Turner, D.W., "Advanced Avionics Fault Isolation System (AAFIS)", Hughes Aircraft Company, Report No. P73-314, August 1973.

3. Bauer, J.E., "Built-In Testing of Large Scale Integrated Circuitry", Naval Air Engineering Center, Philadelphia, PA, October 1973.

4. Bauer, J.E., Benowitz, N., Lee, G.W.K., Yanney, R.M., Morgan, D.C., "Military Microcomputer Self Test Using the Advanced Avionics Fault Isolation Systems (AAFIS)", Hughes Aircraft Company, Report No. P77-93, March 1977.

5. Peterson, W.W., "Error Corrective Codes", MIT Press, 1972.

6. Helness, Karl M., "Implementation of a Parallel Cyclic Redundancy Check Generator", Hewlett-Packard Company Computer Design, March 1974.

7. Benowitz, N., Blandford, D.K., Lee, G.W.K., Shen, J.P., "Final Report AAFIS Design and Demonstration Program", Hughes Aircraft Company, Report No. P75-71, February 1975.

8. Williams, T.W. and Eicheberger, E.B., "A Logic Design Structure for LSI Testing", Proe. 14th Design Automation Conf., New Orleans, (June 1977), pp. 462-468.

APPENDIX I
A CASE STUDY
SCT DEMONSTRATOR UNIT

CONCEPT

For the purpose of further evaluation of the Self-Contained-Test concepts in representative component hardware, a detailed design and simulation of a demonstration unit has been accomplished which will allow physical fault insertion and observation of SCT fault detection and isolation results. Detailed logic design of Self-Contained-Test circuitry for application to the Post Processor of an advanced Hughes radar has been completed. The Post Processor was selected as having logic representative of the many difficult test and isolation conditions in typical applications, including significant control logic and feedback between submodules. This demonstration vehicle contains five submodules totaling 617 ICs, and complete test control (as would normally be implemented in a full module containing Self-Contained-Test) is provided. A complete self-test and isolation to a single module within a group of modules is provided with a high confidence.

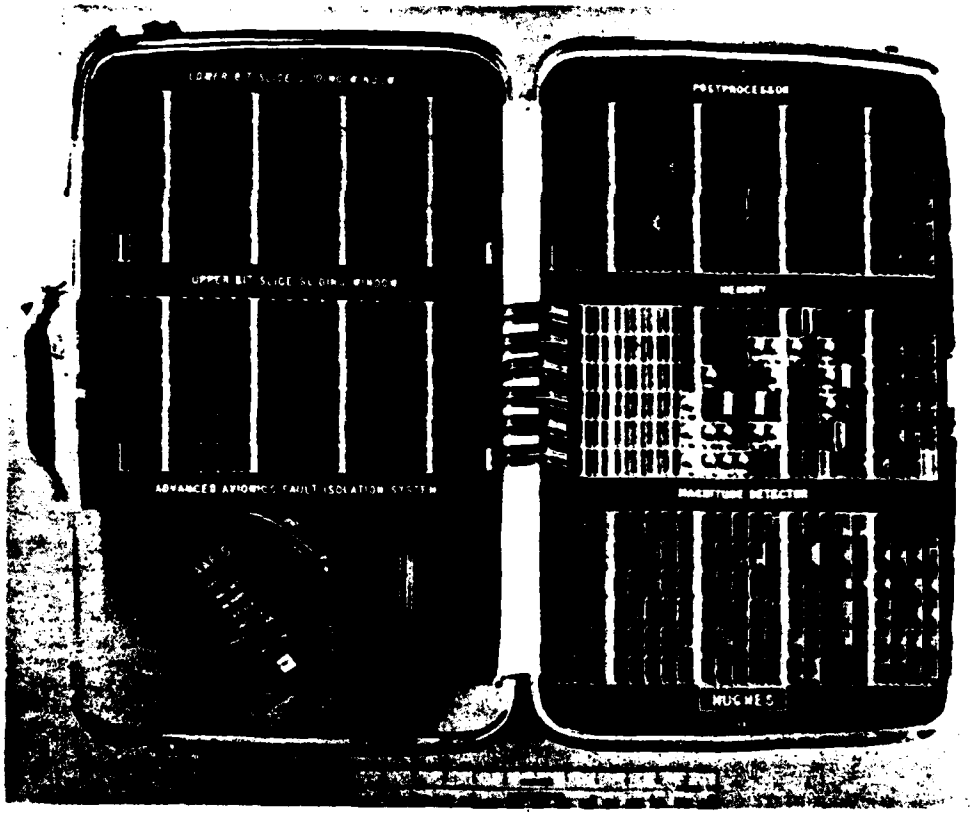


FIGURE A-1 SCT Suitcase Demonstrator

TEST SEQUENCE

See Figure A-2.

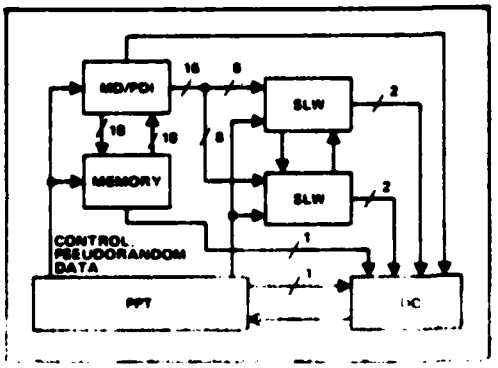


FIGURE A-2. PPT Control Signals

At the start of a test sequence an initialization signal goes from the Demonstration Control (DC) to the Post Processor Timing (PPT) module. This signal then propagates to all other modules. The PPT then receives the clock signal from DC which it divides and decodes to produce the system Main Clock Pulse (MCP) and other clocking signals for the pattern checkers and the multiplexer select lines. The main clock pulse is then used by the PPT to drive the pseudorandom pattern generator which produces test data for entry into the MD/PDI submodule. During the first of the test, the feedback enable lines for the sliding window are disabled and the memory module link from the magnitude detector is blocked. The data (pseudorandom numbers) entering the magnitude detector bypasses the memory and flows to the two sliding window modules. The memory module is then tested by alternately clocking ones and zeros into its shift register memory and monitoring its output. At the end of the first part of the test, the memory and sliding window modules are checked to verify that the test output sequences are as expected. This information is then clocked into the DC logic by signals from the PPT. As the second part of the test begins the gate blocking data flow between the MD/PDI and the memory is enabled to allow pseudorandom numbers to flow through the memory and into the sliding windows. The feedback lines between the sliding window modules are also enabled for testing. The pseudorandom numbers are fed through all the logic again until the PPT generates the test complete signal which clocks either "go" or "no-go" information from each pattern checker into the DC logic. The pattern checker from the memory is excluded from this check since the memory has been checked at at the end of the first test sequence. The sliding windows must be checked twice. The first check verifies the module integrity while the second verifies the feedback integrity. The PPT and MD/PDI are checked only at the end of the test sequence.

CONTROL LOGIC

The demonstration control simulates the initialization of SCT by an external unit, provides a master clock, and controls the display lights and seven segment displays for demonstration. It contains logic not normally included in tactical or Self-Contained-Test hardware and should not be faulted.

The pseudorandom pattern generator, feedback, and mode control SCT logic are placed on the PPT submodule, which also provides normal tactical control to the post processor. The PPT provides control of pattern checkers on all submodules.

LOGIC FLOW DURING TEST

See Figure A-2.

The forward feed logic flow during the demonstrator test mode is from the Post

Processor Timing circuitry to the Magnitude Detector/Post Detection Integration (MD/PDI) submodule. A feedback loop from MD/PDI to and from the PDI Memory submodule is blocked during the early portion of the test when the PPT sends data to both the MD/PDI and Memory. After feedback is enabled, data from the MD/PDI, flows through the memory and back to the MD/PDI.

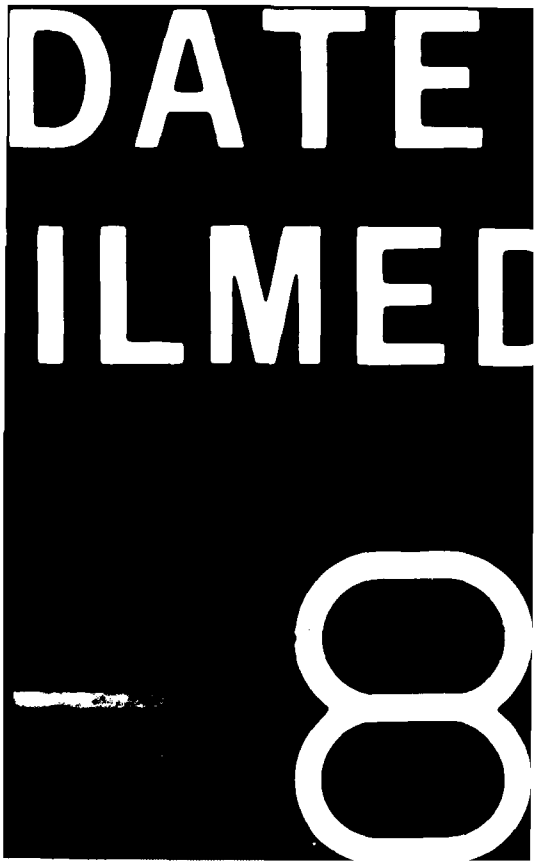
The two identical bit slice sliding window (SLW) submodules receive their data from the MD/PDI outputs. There are feedforward (carriers) and feedback (divided by two functions) between the two-bit slice submodules. Only one direction of logic flow is allowed during each portion of the test.

PATTERN CHECKING LOGIC

The SCT demonstrator utilizes a parallel cycle code which allows all wafer outputs and test points for one input test pattern to be checked in three clocks. An 11-bit code is used by the pattern checker on each submodule except the memory submodule, which uses a nine-bit code. The code checkers are implemented with Hex D registers and full adders.

DISPLAYED RESULTS

A control panel provides a light to indicate when the most recent test has passed. Two additional lights indicate the failure of test #1 (feedforward) or test #2 (feedback). A light is provided for each submodule and will go on when SCT results show that a specific module is faulty. Each submodule provides seven segment displays, which indicate in octal form, the final value of the cyclic code. True or inverted inputs are selected for display input so all zeros will be seen for a correct final code result. The SLW modules have two test result comparisons made (feedforward and feedback). Both of these results are displayed. Displays are blanked on power up, and until the SCT completion timer expires. To allow visual indication of this timer, which provides an error indication if SCT cycle is not completed within a predetermined time, the blanking interval is made about ½ second.


DATE
ILMED
8